Jukka Joensuu

WINDOWS PHONE -SOVELLUS TUOTERAKENTEIDEN ELINKAARIHALLINTAAN CASE: C-CARE

Tietojenkäsittelyn koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

WINDOWS PHONE -SOVELLUS TUOTERAKENTEIDEN ELINKAARIHALLINTAAN CASE: C-CARE

Joensuu, Jukka
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2015
Ohjaaja: Nieminen, Hans
Sivumäärä: 36
Liitteitä: 6

Asiasanat: WP7, Azure, Silverlight, MVVM

Älypuhelimet yleistyvät yhä voimakkaammin ja yritykset käyttävät älypuhelinsovelluksia verkkosovellustensa tukena lisäarvoa antavana toimintana. Esimerkiksi Facebookin tilapäivitykset onnistuvat puhelimen välityksellä, jos tietokoneen käyttömahdollisuus on hankalaa.

Opinnäytetyöhön idea tuli Cieltum Oy:stä, jossa oltiin kiinnostuneita kehittämään Windows Phone -sovellus C-Care -tuotteen tueksi. C-Care on pilvipalveluna toteutettu tuotteen elinkaarihallintasovellus. Tarkoituksena on kehittää älypuhelinsovellus, josta voisi ensisijaisesti merkitä jo etukäteen C-Care -tietokantaan luotuja tehtäviä tehdyiksi ja syöttää tehtäville myös tuntikirjauksia. Windows Phone -käyttöjärjestelmä on luonteva valinta yritykselle, koska sillä oli jo aiemmin vahvat yhteydet Microsoftiin sekä sen tuoteperheeseen.

Työn tuloksena syntyi Windows Phone -puhelimelle ladattava prototyyppisovellus C-Phone sekä dokumentti, johon on kerätty huomioita, rajoituksia ja neuvoja sekä Windows Phone että Silverlight -sovelluskehityksestä ja Windows Azure -käyttömahdollisuuksista.

Windows Phone application for product structures lifecycle management case: C-Care

Joensuu, Jukka
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in business information systems
May 2015
Supervisor: Nieminen, Hans
Number of pages: 36
Appendices: 6

Keywords: WP7, Azure, Silverlight, MVVM

Smartphones become more and more common every day and corporations uses mobile apps to get more options to their web solutions. For example in some scenarios Facebook status updates are easily done with mobile device compared to personal computer.

Idea for this thesis came from Cieltum Oy, which was interested developing Windows Phone mobile application in support of already published C-Care cloud application. Purpose is to develop mobile application, where user can primarily mark previously made tasks ready and create hour inputs to those tasks. Windows Phone operating system was natural choice for company, because it had already strong links to Microsoft and its product family.

The project produced prototype version of C-Phone, Windows Phone application and this document. Document contains collected notifications, restrictions and advices to both Windows Phone and Silverlight software developing and Windows Azure's using capabilities.

LYHENNE- JA KÄSITELUETTELO

Blob	Windows Azure -palvelu, johon sovellus voi tallentaa ja hakea tiedostoja.
C#	Microsoftin .NET standardia varten kehittämä ohjelmointikieli.
EF	Avoimen lähdekoodin tiedonhakumalli .NET sovelluskehitysalustalla. Akronyymi sanoista Entity Framework.
IIS	Microsoftin palvelinohjelmistokokonaisuus Windows-alustoille.
MVC	Model-View-Controller sovellusarkkitehtuurimalli.
MVP	Model-View-Presenter sovellusarkkitehtuurimalli.
MVVM	Model-View-ViewModel sovellusarkkitehtuurimalli.
.NET Framework	Microsoftin kehittämä ohjelmistokomponenttikirjasto.
Pilvipalvelu	Internetissä ulkoisen palvelimentarjoajan palvelimelta ajettava verkkosovellus.
RIA	Rich Internet Application eli rikas Internet sovellus.
RTM	Tarkoittaa sovelluksen markkinoille julkaistua versiota.
SDK	Software Development Kit eli sovelluskehittäjälle ladattavissa olevat luokkakirjastot sovelluskehitysalustalle.
SQL	Lyhenne sanoista Structured Query Language relaatiotietokannan hallintaan.
WCF	Akronyymi sanoista Windows Communication Foundation. Sovelluskehittäjän työkalu, jolla yhdistetään suoraan yhteen sopimattomia luokkakirjastoja toisiinsa.
WP7	Akronyymi Windows Phone 7 -käyttöjärjestelmän nimestä.
WPF	.NET Frameworkin kirjasto, joka muodostaa Windows Vistan ja sen jälkeisten Microsoftin käyttöjärjestelmien graafisen rajapinnan.
WPF/E	Windows Presentation Foundation/Everywhere eli Microsoft Silverlight -teknologian ensimmäisen kehitysversion työnimi.
XAML	Extensible Application Markup Language eli Microsoftin kehittämä XML:ään pohjautuva kieli käyttöliittymien kuvaamiseen.
.XAP	Microsoft Silverlight -projektin automaattisesti pakkaama tiedosto, joka sisältää ajettavan Silverlight -sovelluksen.
XML	Extensible Markup Language eli standardi, jolla kuvataan dokumentissa olevaa tietoa.

SISÄLTÖ

TIIVISTELMÄ

LYHENNE- JA KÄSITELUETTELO

1 JOHDANTO

Olen tietojenkäsittelyalan ammattikorkeakouluopiskelija. Työkokemusta on kertynyt alkaen puolen vuoden opintoihin liittyvästä harjoittelujaksosta koulutusohjelman etenemissuunnitelman mukaisesti. Alan työkokemusta on kertynyt yhdestä firmasta Cieltum Oy:stä ja nyt teen myös opinnäytetyön kyseiselle yritykselle.

Opinnäytetyöni konkreettisena tarkoituksena on tuottaa prototyyppisovellus Windows Phone -käyttöjärjestelmälle. Tarkemmin sovelluksen ensisijainen käyttötarkoitus on työtuntien syöttäminen jo valmiina tietokannassa olemassa olevien tuoterakenteiden huoltotehtäville. Lisäksi näiden tehtävien hallinta, kuten esimerkiksi valmiiksi kuittaaminen, kuuluu projektin tavoitteisiin.

Projektin päämääränä on myös tuoda yritykseen lisää ammattitaitoa älypuhelinsovelluksen kehittämisestä ja kartuttaa Windows Phone -alustan tarjoamia mahdollisuuksia yrityksen C-Care tuoterakenteiden elinkaarihallinnan pilvipalvelun tueksi. Alun perin idea lähti yrityksen johtoportaalta ja koin projektin mielenkiintoisena sekä sopivana itselleni tähän uravaiheeseen, joten tähän tilaisuuteen oli helppo tarttua. Toivon, että tämä myös näkyy työn lopputuloksessa sekä tässä dokumentissa.

Käyn tässä työssä läpi edellä mainittuun päämäärään vaadittavat tekniikat ja pyrin myös kartoittamaan näiden tekniikoiden tarjoamat mahdollisuudet sekä rajoitteet. Koska kyseessä on vasta prototyyppiversio sovelluksesta, pyrin myös erityisesti keskittymään sovelluksen jatkokehitysmahdollisuuksiin sekä saatavilla olevien tekniikoiden tulevaisuuden näkymiin. Esittelen myös yrityksen ja sen tavoitteet, jotta lukija pystyy paremmin ymmärtämään työn etenemisen ja motivaatiot.

Käyn aluksi lyhyesti läpi C-Care -emosovelluksen päätoiminnallisuudet ja sen toteutuksessa käytetyt Microsoftin Silverlight ja Windows Azure -tekniikat, sillä niiden tarjoama rajapinta ja yhtenevyys tarjoavat perustan ja toiminnallisuutta myös tälle projektille. Olen jäsennellyt tekniikkojen läpikäynnin mielestäni loogiseen järjestykseen, jotta lukijan olisi mahdollisimman helppo pysyä ajatuksessa mukana ja työn pystyisi lukemaan alusta loppuun asti ilman ylimääräistä hyppelyä sivujen välillä.

Tekstin ulkoasun pyrin pitämään rentona ja helposti luettavana, mutta samalla myös asiallisena ja tieteellisen kirjoittamisen asettamia raameja kunnioittavana. Työn jälkimmäisellä puoliskolla käydään läpi itse Windows Phone -alustaa ja projektin tuloksena syntynyttä sovellusta sekä sen tarjoamaa lisäarvoa yritykselle, joten innokkaimmillekin lukijoille kärsivällisyyttä.

Työ sisältää myös erikseen sovelluskehittäjille suunnattuja osia. Nämä on otsikoitu erikseen. Myös muut kuin sovelluskehittäjät voivat käyttää tätä työtä esimerkiksi käytettyjen teknologioiden tarjoamien mahdollisuuksien kartoittamiseen. Tällöin voit sivuuttaa nämä otsikot helposti. Nyt toivotan opettavia lukuhetkiä.

2 CIELTUM OY JA C-CARE

2.1 Cieltum Oy

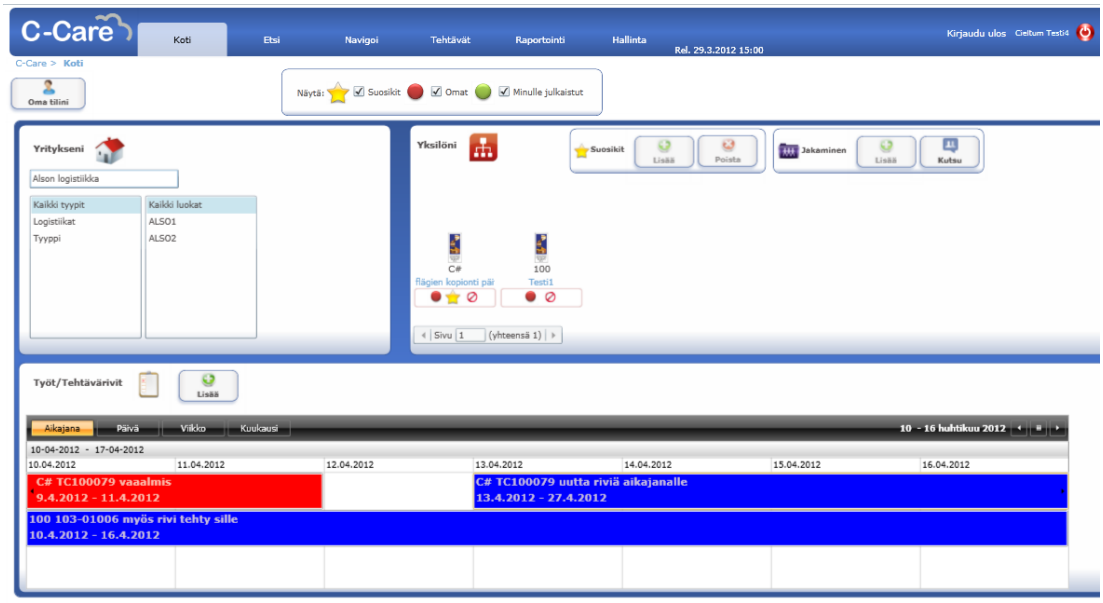
Cieltum Oy on vuonna 2009 perustettu ohjelmistoalan yritys, joka työllistää tällä hetkellä kolme sovellusalan asiantuntijatyöntekijää. Yrityksen toimintaidea perustuu Internet -yhteyden kautta käytettävään C-Care tuoterakenteiden elinkaarihallintaan. Esittelen C-Care -palvelua tarkemmin sekä sovelluksen käytön, käyttötarkoituksen että sen toteutukseen käytettyjen tekniikoiden kautta.

2.2 C-Care

C-Care on tuotteen elinkaarihallintaan suunniteltu pilvipalvelusovellus, joka on toteutettu myöhemmin luvussa 3.1 esiteltävällä Microsoft Silverlight -tekniikalla. Ensimmäinen prototyyppiversio sovelluksesta julkaistiin 2009 keväällä Silverlight 4 -alustalle. Samanaikaisesti Windows Phone -kehityksen kanssa C-Care on päivitetty ajettavaksi Silverlight 5 -versiolla. Pilvipalveluajattelun käytännön toteutus on taas toteutettu luvussa 3.3 esiteltävän Windows Azure -alustan avulla. Lisäksi sovellus on suunniteltu erityisesti yrityskäyttöön, nimenomaan yrityksen sisäisten tuoterakenteiden

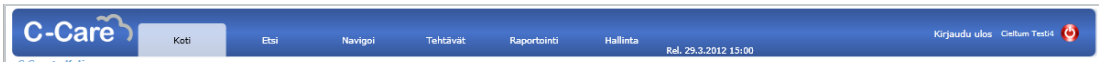
elinkaarihallintaan. Esitän C-Care -toiminnallisuuden vielä tarkemmin kuvakaappausten ja kuvatekstien avulla kuvissa 1 ja 2.

C-Care -etusivun toiminta perustuu vasemmassa yläkulmassa olevaan yritysvalintaan, jonka perusteella oikeassa yläkulmassa sijaitsevaan tuotelistaan latautuvat yrityksen tuotteet. Alaosan reunuksessa taas näkyvät tuotteiden työkohtaiset tehtävät aikajanalla. Tehtäväoliota klikkaamalla voidaan siirtyä tehtävähallinta-näkymään, joka esitellään tarkemmin luvussa 6.2. (Kuva 1.)



Kuva 1. Kuvakaappaus C-Care -etusivusta 10.4.2012.

C-Care -sovelluksen navigointipalkista pääsee etusivulle, jonka päätoiminnot esiteltiin kuvassa 1. Lisäksi navigointipalkista löytyvät ”Etsi”-toiminto, josta pääsee hakemaan yrityksiä, tuotteita tai töitä (työt sisältävät tehtäviä ja ovat linkki tehtävän ja tuotteen välillä). ”Navigointi”-toiminto sisältää tuoterakenteiden hallinnan sekä tuoterakenteeseen perustuvan navigoinnin muualle sovellukseen. ”Tehtävät”-toiminto sisältää tehtävänavigoinnin ja tehtävienhallinnan. ”Raportointi”-valikon kautta avautuvat muun muassa tuntikirjaus, laskutus ja tuntien hyväksyntä toiminnot. ”Hallinta”-sivulla on tietokannan ylläpitoon liittyvien asetusten ylläpito ja lisäys. (Kuva 2.)



Kuva 2. C-Care -sovelluksen navigointipalkki.

3 KÄYTETTÄVÄT TEKNIIKAT

3.1 Microsoft Silverlight

Tässä luvussa käydään läpi Microsoft Silverlight -tekniikan tarjoamat mahdollisuudet sen kehityskaaren avulla. Läpikäynti perustuu etenkin siihen näkökulmaan, jonka olen itse aiemmin kerännyt Silverlight -sovelluskehittäjänä kahden vuoden kokemuksella. Tämä lähestymistapa tarjoaa myös loogisen jatkumon ja selkeän pohjan Windows Phone -sovelluskehittämiselle, kun toteutetaan Microsoft Silverlight for Windows Phone -sovellusta. Luvussa 3.1.1 esiteltävä Silverlight -kehityskaari toimii taas esimerkkinä ja referenssinä Microsoftin kyvystä kehittää sovelluskehitysalustojaan ja Windows Phone -kehitysrajapinnan tulevaisuuden näkymät voi näin ollen osittain perustaa jo aiempaan Silverlight -kehitykseen, koska nämä kuuluvat samaan Microsoftin ekosysteemin rakentamisstrategiaan.

Kuva 3. Microsoft Silverlight -logo.

Käyttäjän näkökulmasta tarkasteltaessa Silverlight on ilmainen selaimelle ladattava liitännäinen, jonka avulla pystyy käyttämään normaaleja selaimen toimintoja monipuolisempia ratkaisuja eli Silverlight -sovelluksia. Sovelluskehittäjän näkökulmasta Silverlight on taas ohjelmointirajapinta, joka tarjoaa mahdollisuuden tuottaa juuri näitä interaktiivisia sovelluksia. Samankaltainen ratkaisu on esimerkiksi joillekin käyttäjille tutumpi Adoben Flash Player. (Microsoft A 2012.)

Silverlight on tuettuna monille eri selaimille kuten Internet Explorer, Mozilla Firefox, Safari sekä Opera ja tuki löytyy myös Windows Phone -alustalle, joten komponenttina Silverlight on laajasti käytettävissä eikä sovelluskehittäjän tarvitse näin ollen erikseen

tehdä omaa sovellusta jokaiselle eri selaimelle. Sovelluksen toimivuus on toki testattava eri selaimilla, mikäli haluaa taata sen laaja-alaisen toimivuuden. Selaimissa on kuitenkin eroavaisuuksia ominaisuuksissa ja toiminnoissa, esimerkiksi suoritusnopeudessa ja tyylitiedostojen tuessa. Tässä suhteessa Silverlight ei kuitenkaan saavuta suurta eroa tai kilpailuetua aiemmin mainittua Flash Playeria vastaan. Tosin se ei jää myöskään huonommaksi vaihtoehdoksi. (Microsoft A 2012.)

3.1.1 Silverlight -teknologian kehityskaari

Otan Silverlight -teknologian kehityskulun vielä tarkempaan tarkasteluun seuraavien alaotsikoiden alla. Tämä toimii referenssinä Microsoftin kyvystä kehittää teknologiaa eteenpäin. Pysyäkseen mukana kehityksessä tulee sovelluksen päivitys uusimpaan kehitysversioon usein perustelluksi valinnaksi. Päivittämisen avulla saakin uusimmat kontrollit käytettäviksi ja seuraavan päivityksen tekeminen helpottuu. Esimerkiksi Silverlight 5 -versioon päivittäminen onnistuu Silverlight 4 -versiota käyttävästä projektista paremmin kuin Silverlight 3 -versiolla toteutetusta sovelluksesta. (Microsoft B 2012.)

3.1.1.1 Silverlight 1.0

Ensimmäinen Silverlight -kehitysversio julkaistiin ladattavaksi 5. syyskuuta vuonna 2007. Version alkuperäinen työnimi oli Windows Presentation Foundation/Everywhere eli lyhennettynä WPF/E. Tämä työnimi taas perustui tarkoitukseen tuoda Microsoftin jo aiemmin tarjoama WPF-tekniikka saataville helpommin Internetin välityksellä. (Järvinen 2007.)

Silverlight 1.0 ei kuitenkaan pystynyt aikanaan tarjoamaan juurikaan kilpailua Adoben kauemmin kehittämälle Flash Player tekniikalle. 1.0 version ominaisuuksiin kuuluivatkin lähinnä video- ja äänitiedostojen käsittely selaimessa eikä aidosti interaktiivisten monipuolisten verkkopalvelusovellusten kehittäminen ollut realistisesti näin ollen vielä mahdollista.

Versio sisälsi tuen sekä Mac että Windows -käyttöjärjestelmille, mutta Linux tuki siitä puuttui ja tätä varten käynnistettiin Moonlight -kehitysprojekti. Yleisesti Silverlight 1.0 -versiota pidettiin kuitenkin huonona ja jopa kehityskelvottomana ratkaisuna. (Järvinen 2007, Udell 2007, Wallaswaara 2007.)

3.1.1.2 Silverlight 2

Silverlight 2 julkaistiin 14.10.2008 ja se tarjosi jo paljon enemmän toiminnallisuutta Silverlight 1.0 -versioon verrattuna. Ehkä tärkeimpänä uudistuksena Silverlight 2 -versioon oli lisätty tuki .NET Frameworkille ja tämän ansiosta Silverlight -sovelluksia on tästä versiosta alkaen voitu kirjoittaa kaikilla .NET ohjelmointikielillä, joista mainittakoon C#, Visual Basic, JavaScript, IronPython ja IronRuby. Tämän uudistuksen myötä Silverlight tavoittikin sovelluskehittäjiä entistä laajemmin. (Lehto 2008; Sneath 2007.)

Versiopäivitys tarjosi myös monia uusia niin sanottuun template-malliin perustuvia kontrolleja. Tämän mallin hyöty tulee esiin kontrollien muokattavuutena yrityksen tarpeisiin sopivaksi ja näin nämä kontrollit sopivatkin hyvin tuotteen jatkokehitykseen. Tarjoten muokattavuuden lisäksi helposti käyttöönotettavan version. Tällaisia kontrolleja ovat esimerkiksi:

- TextBox, johon käyttäjä voi syöttää tekstisyötteen,
- PasswordBox, joka toimii muuten samalla tavalla kuin TextBox paitsi tässä kontrollissa käyttäjä ei näe oman syötteensä sisältämää tekstiä,
- ScrollBar, josta käyttäjä pääsee vierittämään käyttöliittymässä ScrollBar -komponentin sisällä toteutettuja kontrolleja,
- CheckBox, josta käyttäjä voi valita tai poistaa kohteen valinnan,
- RadioButton, joista muodostuvista ryhmistä käyttäjä voi valita elementtejä valituiksi tai ei valituiksi.

Sekä myöhemmät Silverlight -versiot että tässä projektissa käytetty luvussa 3.2 esiteltävä Microsoft Silverlight for Windows Phone perustuvat samankaltaisiin template -malleihin. Näitä malleja on kehitetty eteenpäin ja puhelimelle on luotu uusia, laitteen ominaisuuksia ja käyttötarkoitusta palvelevia, template -mallin kontrolleja. Kuva 4 sisältää tarkemman kuvauksen pelkistetystä Silverlight -käyttöliittymästä, joka sisältää

TextBox, PasswordBox ja Button kontrollit, joiden avulla käyttäjä voi kirjautua sisään palveluun. (Sneath 2007.)

Kuva 4. Silverlight 2 -käyttöliittymä. (Gren 2008.)

Silverlight 2 -versio sisälsi muun muassa RIA-kehitystyökalun, jonka avulla Silverlight -sovellus pystyy ajamaan samanaikaisesti päällekkäisiä prosesseja. Tästä huolimatta Silverlight 2 oli yhä Flash Playerin kehitystä jäljessä. Etenkin sen toimivuus muilla kuin Internet Explorer -selaimella aiheutti kysymysmerkkejä. Kuten myös käyttäjien halu asentaa Silverlight -liitännäistä selaimelle, kun Flash Player liitännäinen löytyi jo usean käyttäjän järjestelmästä. (Lehto 2008.)

3.1.1.3 Silverlight 3

Heinäkuussa 2009 julkaistu Silverlight 3 -päivitys ei ollut yhtä suuri edistysaskel kuin toinen versio verrattuna ensimmäiseen. Tämä johtui osin ensimmäisen version niukkuudesta ja toisen version mukanaan tuomasta suuresta päivityksestä. Silverlight 3 -version voidaan sanoa tuoneen toiseen versioon täydennystä, kun toinen versio oli selkeä uudistus ensimmäiseen versioon nähden.

Tässä päivityksessä Microsoft oli keskittynyt muun muassa liitännäisen media- sekä grafiikkaominaisuuksiin. Mediaominaisuuksista mainittakoon versioon lisätty tuki H.264-videoille sekä AAC-audiolle. Graafisen toteutuksen mahdollisuudet sekä yleisestikin sovelluksen interaktiivisuus ja kehitysmahdollisuudet taas paranivat muun muassa 60 uuden kontrollin ja sovelluksen ulkoasuun liittyvien uudistusten, kuten 3D-grafiikka prosessoinnin julkaisuilla. (Heikniemi 2009.)

Silverlight 3 -versioon lisättiin myös selaimen ulkopuolinen tuki, joka tarkoitti että Silverlight -sovellusta voitiin ajaa työpöydältä ilman Internet-yhteyttä. Ensisijaisesti Silverlight oli kuitenkin yhä suunniteltu ja tarkoitettu verkkosovellus käyttöön eikä tähän lähtökohtaan ollut tarkoitus tuoda muutosta. Parantuneiden grafiikkaominaisuuksien myötä kehittäjille avautui toisaalta mahdollisuuksia luoda näyttävämpiä käyttöliittymätoteutuksia. (Heikniemi 2009.)

Silverlight 3 -versioon lisättiin myös tuki kosketusnäytöille sekä monikosketusohjaukselle. Tällä Microsoft pyrki vastaamaan Applen aloittamaan kehitykseen kosketusnäyttöteknologian yleisempään käyttöön ja näin Silverlight 3 olikin taas enemmän ajan hermolla oleva verkkototeutusmahdollisuus kuin teknologian edeltävät päivitykset. Silverlight -teknologiaa kritisoitiin toki edelleen Microsoftin sisäisenä toteutuksena ja Microsoftin ulkopuolisilla tahoilla epäiltiin tämänkin päivitysversion kilpailukykyä Adobe Flash Playeriin sekä kehittyvään, avoimeen ja käyttöjärjestelmäriippumattomaan HTML5-standardiin. (Lehto 2009.)

3.1.1.4 Silverlight 4

Huhtikuussa 2010 julkaistu Silverlight 4 -versio toi mukanaan tukun uudistuksia. Kuten parannetun kielituen, lisää tulostusominaisuuksia sekä tuen web-kameralle ja mikrofonille. Lisäksi tämä päivitys toi parannuksia sovelluksen suoritusnopeuteen ja suorituskykyyn. Silverlight pystyi nyt muun muassa hyödyntämään aiempaa paremmin näytönohjaimien piirto-ominaisuuksia. Perusajatus tarjota interaktiivisen verkkosovelluksen kehitysympäristö ei juuri kehittynyt monipuolisemmaksi, mutta tämän päivityksen avulla kehitetyt sovellukset toimivat entistä paremmin. (Foley 2010; Räihä 2010.)

Käytettävyydeltään Silverlight -alustalle tuli uusina ominaisuuksina hiiren parempi hyödyntäminen. Tämä tarkoitti esimerkiksi hiiren rullan käyttöä kontrollien sisällä sekä kohteen aktivointia hiiren painikkeen valinnan ajaksi. Päivitykseen lisättiin myös leikepöydältä kopioiminen ja liittäminen. Versio oli selkeä edistysaskel. Kehitettävää toki vielä löytyi esimerkiksi suorituskyvyssä, vaikka sitä olikin parannettu edelliseen

versioon verrattuna. Mainittakoon vielä, että luvussa 3.2 esiteltävä sovelluskehitystekniikka, jolla itse projektin lopputuotteena syntynyt sovellus toteutettiin eli Windows Phone 7 perustuu tähän Silverlight -versioon. (Masalin 2010.)

3.1.2 Sovelluskehittäjän työkalut

Sovelluskehittäjälle Silverlight -kehitykseen tarjolla olevia työkaluja ovat muun muassa Microsoft Visual Studio, Expression Blend, Expression Design, Expression Web ja SketchFlow. Näistä Visual Studio on suunniteltu lähinnä ohjelmoijan työkaluksi ja sillä voidaan luoda Silverlight -sovellukseen monipuolisia dataliikennemalleja. Expression Studion Blend, -Design ja -Web -ohjelmat ovat taas käyttöliittymän toteutukseen tarkoitettuja työkaluja. Silverlight -käyttöliittymän toteutus onnistuu myös Visual Studiolla, mutta Expression Studio tarjoaa mahdollisuuden käyttöliittymän toteuttamiseen vähemmän ohjelmointiin perustuvasta graafisemmasta näkökulmasta. (Järvinen 2007.)

Expression Studio ja Visual Studio -toteutukset voidaan myös liittää yhteen versiohallintaohjelmiston avulla ja näin näiden ohjelmien ominaisuudet voidaan yhdistää sovelluskehitysympäristöksi. SketchFlow -ohjelma on taas työkalu käyttöliittymän suunnitteluun ja sillä voi luoda pohjia Silverlight -toteutuksille. Näitä luonnoksia voi käyttää lähtökohtana itse sovelluksen toteutukselle. (Järvinen 2007; Lehto 2008.)

3.2 Windows Phone

Windows Phone on Microsoftin ratkaisu älypuhelinmarkkinoille. Windows Phone -puhelimissa sovelluksissa käytetään Windowsista tuttua XML-pohjaista Silverlight -käyttöliittymäkirjastoa ja -puhelinversiolla kehitetyt versiot julkaistaan Windows Phone -sovelluksien omassa App Martketplace -sovelluskaupassa, joka on käytettävissä vain Windows Phone -käyttöjärjestelmän sisältävissä puhelimissa. Toisaalta App Marketplac on saatavilla jokaisessa Windows Phone -puhelimessa. (Järvinen 2012, 12.)

Kuva 5. Windows Phone -käyttöjärjestelmän logo.

Puhuttaessa Metrosta Windows Phone -sovelluskehityksessä tarkoitetaan sovelluksen käyttöliittymää ja Microsoftin kehittämää käyttöliittymästandardia työnimeltään Metro. Metroa käytetään myös Windows 8 -käyttöjärjestelmässä eikä se ole pelkästään puhelinalustalle optimoitu käyttöliittymä, vaan kuuluu enemmänkin Microsoftin ekosysteemi strategiaan. Windows Phone -alustaa käydään vielä tarkemmin läpi käytännön esimerkkien kautta alkaen luvusta 4. (Järvinen 2012, 54.)

3.3 Windows Azure

Windows Azure on pilvipalvelujärjestelmä, jota ajetaan toisiinsa yhdistetyistä Microsoft Windows Server -pohjaisista tietokoneresursseista, jotka sijaitsevat resurssikeskuksissa eri puolilla maapalloa. Käytännössä näiden resurssien kautta palvelun asiakas voi hallita sovelluksia sekä ylläpitää tietokantaa Internetissä ilman omia palvelimia. (Betts 2010, 1-2.)

Windows Azure voidaan jakaa erikseen laskenta- (*Windows Azure Compute*) ja tallennusympäristöön (*Windows Azure Storage*). Pilvipalvelujärjestelmän pohjalle on rakennettu alusta (*Windows Azure Platform*), johon kuuluu erilaisia komponentteja, kuten Windows Azure AppFabric, SQL Azure ja itse Windows Azure. Tämän projektin kannalta oleellisimpia näistä komponenteista ovat SQL Azure ja Windows Azure web-rooli palvelut, joista kerron vielä tarkemmin. (Betts 2010, 1-2.)

Kuva 6. Windows Azure -tuotteen tunnistat tästä logosta.

3.3.1 Laskentaympäristö

Windows Azure pilvipalvelujärjestelmän isännöimät resurssit sisältävät kahdenlaisia instansseja:

- työskentelijärooli (*worker role*),
- web-rooli (*web role*).

Nämä rooli-instanssit toimivat rajapintana Windows Azure:n ja sovelluksen välillä siten, että työskentelijäroolit isännöivät koodia ajaen jatkuvasti pitkäkestoisia ei interaktiivisia tehtäviä. Web-rooli taas hyväksyy kehitetystä sovelluksesta päin tulevia HTTP (*Hypertext Transfer Protocol*) tai salattuja HTTPS (*Hypertext Transfer Protocol Secure*) -kutsuja. Web-roolin voi yhdistää mihin tahansa kehitysympäristöön, joka toimii IIS:n kanssa. Tällaisia ovat esimerkiksi ASP.NET, Microsoft Silverlight, Windows Communication Foundation (*WCF*) sekä PHP. (Betts 2010, 1-5.)

Molemmat roolit voivat tehdä ulospäin suuntautuvia TCP-yhteyksiä ja avata avoimia päätepisteitä sisäänpäin tuleville kutsuille. Näin Windows Azure -palvelusta tieto kulkee molempiin suuntiin. Käytännössä Windows Azure -pilvessä pyörivässä sovelluksessa täytyy olla joko työskentelijä- tai web-rooli, mutta se voi sisältää myös molempia. Esimerkiksi web-rooli voi vastaanottaa tulevia käskyjä ja siirtää ne työskentelijärooliin prosessoitavaksi, vaikka WCF-palvelun välittämänä. Tässäkin projektissa käytettyyn WCF-palveluun palataan vielä tarkemmin luvussa 4.2.2.1. (Betts 2010, 3-6.)

3.3.2 Tallennusympäristö

Windows Azure tarjoaa skaalautuvia tiedon tallennuspalveluja, joita hallitaan Windows Azure -verkkopalveluun luodun käyttäjätilin kautta (löytyy osoitteesta http://manage.windowsazure.com). Yhdelle tilille voi tallettaa tietoa 100 teratavun verran, mutta tilejä voi tehdä useita. Näitä tiedon tallennuspalveluja on neljän erityyppisiä:

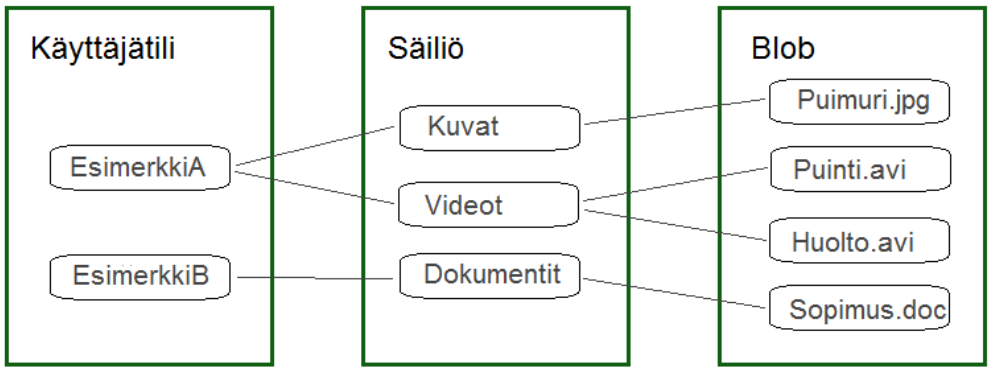
- blob,
- levy (*drive*),
- taulu (*table*),
- jono (*queue*). (Betts 2010, 6-7.)

Esittelen Windows Azure -tallennuspalveluista tarkemmin vain tämän projektin käyttämän blob -palvelun luvussa 3.3.2.1. Lyhyesti mainittakoon, että Windows Azure -taulupalvelu sisältää tietokantatauluja, jotka eroavat tyypillisistä relaatiotietokannan tauluista siten, ettei niihin ole toteutettu relaatioita eikä niissä ole sisäistä mallia (*schema*) ja näin tiedon haku on nopeampaa kuin esimerkiksi SQL Azure -tietokannasta. Tiedon linkittäminen taulujen välillä on samalla silti hankalampaa ilman suoria relaatioita. (Betts 2010, 11.)

3.3.2.1 Blob -palvelu

”Blobbiin” voit tallentaa esimerkiksi videoita, dokumentteja, kuvia tai koodia siten, että käyttäjätilin alle luodaan säiliöitä (*container*) ja säiliöiden alle voidaan tallentaa tiedostoja tai luoda lisää säiliöitä. Säiliöiden ja tiedostojen määrää ei ole erikseen rajoitettu ja ainoa vastaantuleva raja on käyttäjätilin 100 teratavun tallennuskapasiteetti. Sekä säiliöt että tiedostot voivat lisäksi sisältää 8 kilotavua metatietoa muodossa nimi/arvo (KeyValuePair() -metodi). (Betts 2010, 7-8.)

Jokaiselle blobissa olevalle tiedostolle luodaan viitattava URL-osoite, joka on muotoa http://<käyttäjätilin nimi>.blob.core.windows.net/<säiliön nimi>/<tiedoston nimi>. Tämän osoitteen avulla sovellus voidaan yhdistää tallennettuihin tiedostoihin. Kuviossa 1 on kuvattu tämä malli. Käyttäjätilillä voi olla monta säiliötä ja säiliön alla voi olla yksi tai useampia tiedostoja tallennettuna. Lisäksi käyttäjällä voi olla useita käyttäjätilejä. Kaaviosta voi lukea myös tiedoston tallennuspaikkana toimivan URL-polun. Esimerkiksi "Huolto.avi" -video löytyy osoitteesta http://EsimerkkiA.blob.core.windows.net/Videot/Huolto.avi. (Betts 2010, 8.)



Kuvio 1. Blob -rakennekaavio.

Tietoturvamielessä blob -palvelussa tiedostojen suojaus tapahtuu säiliöiden kautta. Oletusasetuksena tiedostoihin pääsee käsiksi vain salatulla avaimella. Tätä voidaan muuttaa siten, että jokainen säiliö voidaan asettaa julkiseksi (*public*) tai yksityiseksi (*private*). Tämä vaikuttaa siten, että yksityisten säiliöiden sisältämien blob -tiedostojen lukeminen (*read*) tai muokkaus (*write*) vaatii verkkopalvelun käyttäjätunnukset. Julkisia blob -tiedostoja taas pääsee lukemaan ilman käyttäjätunnuksia ja näin blob -palveluun tallennettua tietoa voi jakaa yleisesti internetissä ilman, että käyttäjä pääsee muokkaamaan tietoa. (Betts 2010, 8; Chappell 2010, 19)

3.3.3 SQL Azure -tietokanta

SQL Azure on käytännössä pilvipalveluna toimiva tietokantapalvelin. Se on pääasiallisesti samanlainen kuin SQL Server, paitsi että sen ajo suoritetaan Windows Azure -pilvipalvelun kautta ja näin ollen palveluntarjoajan ei tarvitse itse huolehtia esimerkiksi fyysisistä palvelimista tai varmuuskopioinnista samalla tavalla kuin käytettäessä SQL Server -tietokantaa. (Betts 2010, 14; Chuvyrov, 2011, 37-38.)

SQL Azure -tietokantaan yhdistäminen onnistuu eri tekniikoilla kuten esimerkiksi PHP, ADO.NET tai ODBC (*Open Database Connectivity*). Nämä tekniikat ovat samoja kuin SQL Serverissä, joten valmiin tietokannan siirtäminen SQL Serveristä SQL Azure -tietokannaksi vaatii vain liitäntätunnisteen (*connection string*) vaihtamista. (Betts 2010, 14.)

Windows Azure -pilvessä olevan sovelluksen voi yhdistää SQL Azure -pilvessä olevaan tietokantaan esimerkiksi web-roolissa olevan WCF-palvelun kautta. Tähän tekniikkaan palaamme luvussa 4.2. Kuitenkin vaikkei sovellusta ajettaisikaan Windows Azure -pilvipalvelun kautta, niin se voi siltikin käyttää SQL Azure -tietokantaa ja syöttää sinne tietoa. Näin esimerkiksi työpöytäsovellus pystyy hyödyntämään osittain Windows Azure -palveluja. (Betts 2010, 1, 14.)

3.3.4 Laskutusmallit ja tuoteidea

Jokaista Windows Azure -pilvipalvelujärjestelmän pääkomponenttia laskutetaan erikseen. Laskutus tapahtuu Microsoft Online Services -verkkopalvelun kautta ja käyttääksesi maksullisia Windows Azure -palveluita tarvitset tunnukset palveluun. On huomattava että vasta maksettuasi tarvitsemasi palvelut voit siirtää verkkosovelluksen tuotantoon. Käytännössä tämä tarkoittaa säilyvän ja omavalintaisen domainin käyttömahdollisuutta. (Betts 2010, 16.)

Windows Azure -käyttö laskutetaan siten, että instanssin ollessa tuotannossa sitä laskutetaan tunneittain. Instansseja voi ostaa yhdelle sovellukselle myös useampia, mutta minimissään sovellus pyörii yhdellä instanssilla. Kuitenkin mitä enemmän instansseja, sitä enemmän laskentatehoa sovelluksella on käytössä. Windows Azure -tallennuspalveluista taas laskutetaan käytön perusteella. Esimerkiksi, jos blobissa on ollut yhden gigatavun kokoinen tiedosto kahden päivän ajan, siitä laskutetaan 2 gigatavua kuukaudessa. (Microsoft C, 2011.)

SQL Azure -verkkopalveluun voi taas luoda joko 1 GB, 10 GB tai 50GB tallennuskapasiteetin omaavan tietokannan. Laskutus tapahtuu tässä myös maksimikapasiteetin mukaan, toisin kuten tallennuskapasiteetista. Tällöin käyttäjä maksaakin oikeastaan enemmän potentiaalisesta käyttötarpeesta. (Betts 2010, 14.)

3.3.5 Riskitekijät ja riskinhallinta

Kuten muissakin pilvipalveluissa myös Windows Azure -palvelussa asiakas on riippuvainen palveluntarjoajasta. Mikäli Microsoft lopettaisi Azure -alustan tuen tai kehittämisen tuottaisi se verkkosovellukselle ongelmia ja pahimmassa tapauksessa koko sovellus jouduttaisiin toteuttamaan uudestaan alusta alkaen. Entä miten Microsoft onnistuu torjumaan tietoturvauhat. (Heikniemi 2009.)

Käytännössä laittamalla sovelluksen pyörimään Windows Azure -alustalle, myös palvelimien tietoturva on ulkoistettuna palveluntarjoajalle. Tällöin Windows Azure -pilvipalvelujärjestelmää käyttöönotettaessa on tiedostettava nämä riskit ja luotettava Microsoftin kykyyn palveluntarjoajana myös tulevaisuudessa. Windows Azure -asiakkaan tuleekin ja on pakko hyväksyä nämä riskit. (Heikniemi 2009.)

4 SOVELLUKSEN SUUNNITTELU

4.1 Sovellusarkkitehtuuri

Ennen sovelluksen kehittämisen aloittamista oli oleellista valita käytettävä sovellusarkkitehtuuri, jonka mukaan sovellus rakennetaan. Täten sovelluksesta saa helpommin ylläpidettävän sekä selkeästi luettavan ohjelmoijan näkökulmasta. Tässä osiossa käyn vielä tarkemmin läpi tätä valintaa, mahdollisten vaihtoehtojen ja itse syntyneen ratkaisun kautta.

4.1.1 MVVM-arkkitehtuurimalli

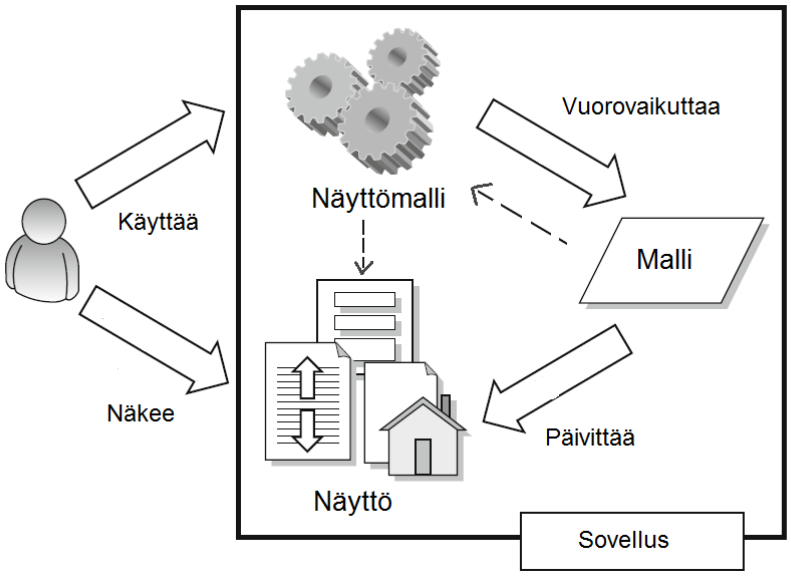
Ensimmäisenä mahdollisista vaihtoehdoista tuli mieleen myös C-Care -sovelluksessa käytetty MVVM-arkkitehtuurimalli. Sekä Silverlight että Windows Phone -sovelluskehityksessä on yleistynyt MVVM-arkkitehtuurimallin käyttö. Tässä mallissa on kolme pääelementtiä:

- malli (model),

- näyttö (view),
- näyttömalli (viewmodel).

Toiminnallisuudeltaan nämä jakautuvat siten, että malli sisältää sovelluksen käyttämän tiedon. Näyttö taas näyttää mallin sisältämän tiedon sovelluksen käyttäjälle ja sovelluskehittäjän näkökulmasta näyttöihin rakennetaan itse sovelluksen ulkoasu. Näyttömalli on eräänlainen ohjain näytön ja mallin välillä. Sen tehtäviin kuuluu esimerkiksi sellaisten tapahtumien käsittely, jotka vaikuttavat joko sovelluksen tiedon käsittelyyn tai sovelluksen ulkoasun päivittämiseen esimerkiksi muuttuneen tiedon tai käyttäjän antaman komennon mukaan. (Chuvyrov, 2011, 36.)

Edellä mainitun jaottelun mukaan saavutetaan selkeä erottelu sovelluksen logiikan, käyttöliittymän sekä tietojenkäsittelyn välille. Kuvio 2:ssa käyttäjä käyttää sovelluksen näyttömallia, josta sovelluksen sisältämä tieto päivitetään malliin. Sitten malli päivittää tiedon joko suoraan tai näyttömallin kautta näytölle, josta käyttäjä näkee tapahtuneen muutoksen. Esimerkiksi listamuuttujan tapauksessa näytöllä on listakontrolli, jonka listadata on näyttömallissa, mutta yksittäinen tietokantaolioon sidottu listaolio on mallissa. (Chuvyrov 2011, 36-37.)



Kuvio 2. MVVM-arkkitehtuurimallia havainnoiva kaavio. (Chuvyrov, 2011, 36-37.)

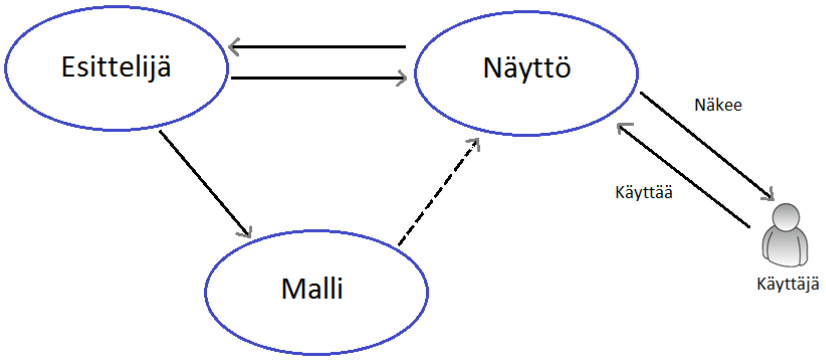
4.1.2 MVP-arkkitehtuurimalli

Vaihtoehdon edellä mainitulle MVVM-mallille tarjoaa myös Microsoftin sovelluskehitysalustoilla yleisesti käytetty MVP-arkkitehtuuri. Molemmat näistä arkkitehtuurimalleista perustuvat MVC-arkkitehtuuriin ja MVP-arkkitehtuuri perustuu myös kolmeen pääelementtiin kuten MVVM-arkkitehtuuri. Yhdenmukaisuus on havaittavissa myös näiden elementtien nimeämisissä:

- malli (model),
- näyttö (view),
- esittelijä (presenter).

Malli ja näyttö toimivat samoin kuin edellä kuvatussa MVVM-arkkitehtuurimallissa ja eroavaisuudet tulevatkin esiin itse käyttäjän roolissa sovelluksen käytössä sekä sovelluskehittäjän työn kannalta esittelijän ja näytön toteutuksessa. Tarkemmin MVP-mallissa käyttäjä käyttää suoraan näyttöä, johon on sidottu käyttäjän antamat komennot sovellukselle, joiden tulokseen esittelijä reagoi. MVVM-mallissa komentojen suorittaminen on taas mahdollisimman paljon sidottuna näyttömallin toteutuksessa. (Sæther, 2011; Kuvio 3.)

Kuvio 3 havainnollistaa datan muutoksen kulkua MVP-mallissa. Käyttäjä on suoraan tekemisissä vain näytön kanssa, joka vastaanottaa käyttäjän syötteen ja reagoi siihen esittelijän kautta. Lopulta käyttäjä näkee näytöltä esittelijän kautta syötteen perusteella tehdyt muutokset. Samalla esittelijä käyttää tarpeen mukaan mallia muutosten tallentamiseen tietokantaan. Näyttö myös näyttää mallissa sijaitsevat tietokantaoliot, mutta tämä tapahtuu epäsuorasti esittelijän kautta. (Sæther, 2011.)

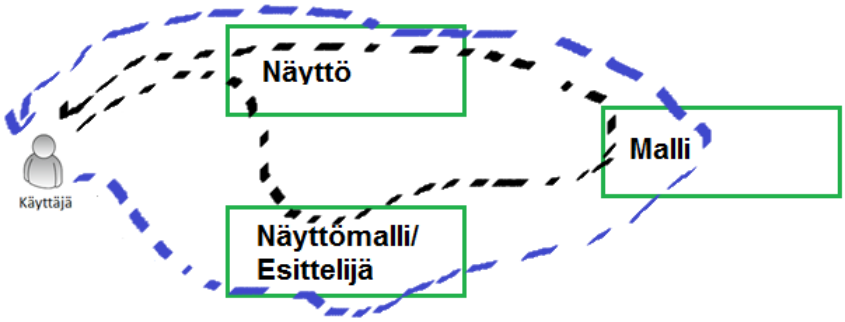


Kuvio 3. MVP-arkkitehtuurimallia havainnoiva kaavio. (Sæther, 2011.)

4.1.3 Käytettävän toteutuksen valinta

Käytettävä sovellusarkkitehtuuri C-Phone -projektiin valittiin emosovelluksessa tehdyn valinnan perusteella. Koska MVVM-arkkitehtuuri soveltuu myös Windows Phone -sovelluskehitysalustalle, toteutuksen ylläpidettävyyden kannalta tässä kohtaa päädyttiin samaan ratkaisuun. Näin jo aiemmin C-Care -tuotteen parissa työskennellyt työryhmä pystytään tuomaan helpommin mukaan myös C-Phone -kehittämiseen tarpeen vaatiessa ja tämä on tärkeää etenkin tehokkaan jatkokehityksen kannalta.

Käytännössä valinta ei kuitenkaan ole näin yksioikoinen. Vaikka käytössä onkin MVVM-arkkitehtuuri, niin todellisuudessa on jouduttu käyttämään kompromissiratkaisua, jossa hyödynnetään myös MVP-arkkitehtuurin tuomia lisäominaisuuksia. MVP-arkkitehtuurin käyttöliittymälähtöisyys avaa sovelluksen käyttökokemukselle uusia ovia ja näin joissain tapauksissa on perusteltua MVP-mallin mukainen näytön käsittely MVVM-mallissa, vaikka sovelluksen tiedonkäsittely perustuukin MVVM-arkkitehtuuriin. Syntynyttä ratkaisua voisi kutsua esimerkiksi MVVM(P)-arkkitehtuurimalliksi. (Kuvio 4.)



Kuvio 4. Toteutunut MVVM(P)-arkkitehtuurimalli.

Kuvio 4 havainnollistaa kuinka syntyneessä toteutuksessa käyttäjä käyttää sovellusta, joko MVP-mallin mukaisesti (musta viiva) käyttämällä näyttöä. Päivittämällä näyttömalliin sovelluksen sisältämä tieto ja tietokantapäivitys mallin kautta, käyttäjä näkee kuittauksen näytöltä. Vaihtoehtoisesti käyttäjä voi, sovelluksen kehitys valinnasta riippuen, käyttää MVVM-mallin mukaista reittiä eli käyttää näyttömallin sisältämää ko-

mentoa, joka päivittää tiedon mallin kautta näytölle. Itse käyttäjä ei sovellusta käyttäessään huomaa eroa, mutta tiedon kulun ja sovelluksen kehittämisen kannalta tämä tieto on oleellista kehittäjälle. (Kuvio 4.)

4.2 Windows Phone -sovelluksen yhdistäminen SQL Azure -tietokantaan

Luvuissa 3.2 ja 3.3 on jo tutustuttu tarkemmin Microsoftin Windows Phone -sovelluskehitysrajapintaan ja Windows Azure -alustaan erikseen. Tämän osion tarkoituksena on ymmärtää tarkemmin, miten näiden palveluiden yhdistäminen tapahtuu ja mitä mahdollisuuksia tällä liitoksella saavutetaan. Tässä osiossa oletetaan, että SQL Azure -tietokanta sekä WP7-sovellus ovat jo toteutettuina ja huolehditaan vain rajapintojen ydistämisestä. Dokumentissa ei käydä erikseen läpi SQL Azure -tietokannan perustamista, koska dokumenttiin liittyvässä projektissa sellainen löytyi jo valmiina. Lähteistä löytyvästä Chuvyrovin ja Leen kirjasta löytyy kuitenkin englanninkielinen esimerkki aiheesta alkaen sivulta 38.

4.2.1 Entity Framework

Ensiksi on tärkeää ymmärtää, että sovellus ei ota suoraan yhteyttä tietokantaan. Tähän käytetään ADO.NETin Entity Framework (EF) -tekniikkaa, joka on työkalu olioiden välisten suhteiden kartoitukseen. Käytännössä EF:n avulla pystyt luomaan automaattisesti olioita (jotka perustuvat tietokannassa oleviin tauluihin) sekä kartoittamaan näiden olioiden väliset riippuvuudet. Ilman EF:n käyttämistä sovelluskehittäjän pitäisi ohjelmoida nämä toiminnallisuudet itse, joten EF:n käyttö on perusteltua ajan säästämiseksi. Sen kautta pystytään suorittamaan myös tarpeelliset tietokantaolioiden lisäys-, luku-, päivitys- ja poistotoimenpiteet. (Chuvyrov, 2011, 38.)

4.2.2 Web-rooli

Kun SQL Azure -palvelimella on olemassa valmis toimiva tietokanta, tarvitaan web-rooli ottamaan siihen yhteys. Lisäksi palvelun tulee olla sellainen, että WP7-rajapinnan tuetut luokkakirjastot voivat lukea sitä. Web-roolin onkin tarkoitus tarjota

ohjelmointirajapinta yhteydenottoon molempiin suuntiin. Tämä tavoite saavutetaan luomalla WCF-palvelu (*Windows Communication Foundation -service*), jota isännöidään web-roolin kautta. Käytännössä web-rooli on itse käytettävä web-palvelin, joka sisältää ohjelmoitavia komponentteja. (Chuvyrov, 2011, 50)

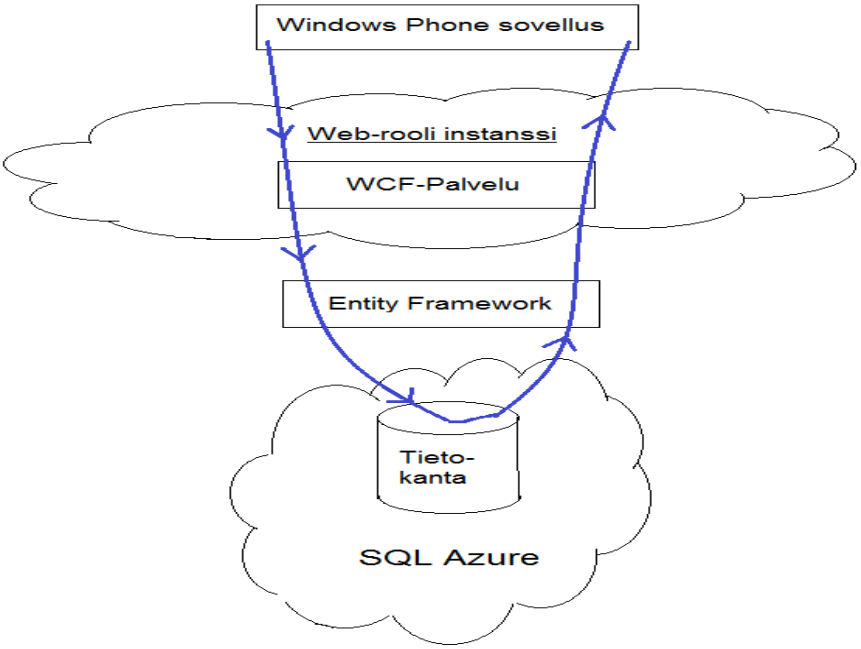
4.2.2.1 WCF-palvelu

WCF-palvelun tehtävä on tarjota ohjelmointirajapinta, jota kutsumalla sovellus voi suorittaa tarvittavat tietokantaoperaatiot. Tietokantayhteys WCF-palveluun on kuitenkin mahdotonta ottaa suoraan, joten tätä varten luodaan luvussa 4.2.1 kuvattu Entity Framework, jota WCF-palvelu käyttää. WCF-palvelun perustamiseen tarvitaan ensin Windows Azure -projekti, jonka kautta hallitaan rooli-instanssia, johon WCF-palvelu liitetään. (Chuvyrov, 2011, 50-51.)

Silverlight for Windows Phonen WCF-ominaisuudet perustuvat Silverlight 3:en WCF -rajapintaan muutamilla lisäyksillä. Silverlight -kehittäjien tulee kuitenkin huomioida, ettei WP7-rajapinnan tarjoama tuki WCF-rajapinnalle, ole yhtä viimeistelty kuin web-kehityspuolella. Tämä seikka tulee myös ottaa huomioon suunniteltaessa ja toteutettaessa WCF-palvelua WP7-sovelluksen käytettäväksi. Tämä ongelma ilmeni myös C-Phone -projektissa, kun C-Care -sovelluksen käyttämään WCF-palveluun ei voitu ottaa yhteyttä WP7-projektista ja uuden palvelun luominen oli näin ollen välttämätöntä. (Cameron, 2011, 251.)

4.2.3 Windows Phone -projekti

WP7-projektista yhteys muodostetaan, joko paikallisesti tai palvelimelta ajettavana olevaan SOAP- tai REST-palveluun lisäämällä palvelurajapinta haetusta osoitteesta. Kuvio 5 kuvaa WP7-sovelluksen yhteydenottoa SQL Azure -palvelimella olevaan tietokantaan. Tähän väliin tarvitaan web-roolin kautta ajettava WCF-palvelu, joka ottaa yhteyden Entity Frameworkin kautta itse tietokantaan. Datan päivitys sekä tietokantahaut tapahtuvat saman WCF-palvelun kautta. (Cameron, 2011, 251.)



Kuvio 5. WP7-sovelluksen ja SQL Azure -tietokannan yhdistäminen. (Chuvyrov, 2011, 38, 50.)

4.2.4 SQL Azure -tietokanta

SQL Azure -palvelimelle tulee vielä lisätä palomuuripoikkeukset sekä oman koneesi lähettämälle IP-osoitteelle (jotta voit testata WCF-palvelua paikallisesti) että luomallesi web-roolin käyttämälle IP-osoiteavaruudelle. Lisäksi tietokannan tulee sallia Microsoftin palveluiden yhteydenotto tietokantapalvelimelle. Näillä asetuksilla tietokantaa voi käyttää WP7-puhelimessa olevasta sovelluksesta, kun puhelin on kytkettynä Internetiin. (Chuvyrov, 2011, 43-45, 50.)

4.3 Käyttäjänhallinta

Sekä C-Care että C-Phone -ohjelmassa tulee vielä ongelmaksi tiedon henkilökohtaisuus. Sekä yrityskohtaisissa että yhden käyttäjän tarpeita varten suunnitelluissa pilvipalveluissa tulee usein pystyä rajaamaan tiedon muokkaus vain osalle käyttäjätileistä. Esimerkiksi Cieltum Oy:n tuoterakenteiden elinkaarihallintasovellusten käyttötapauksessa tietojen muokkaus-, luku- ja päivitysoperaatiot tulee näkyä vain tuotteen omis-

tajalle sekä niille henkilöille, joille omistaja haluaa näyttää nämä tiedot. Oman sisäänkirjautumispalvelun rakentaminen on taas kallista eikä se ole usein oleellista yrityksen oman tuoteidean eteenpäinviemisessä, joten näin sellaisen rakentamiseen käytetty aika on käytännössä ylimääräistä työtä.

Lisäksi käyttäjän kannalta on helpompaa, jos hän voi käyttää jo mahdollisesti valmiina olevia vanhoja tunnuksiaan sovellukseen kirjauduttaessa. Tätä varten Windows Live -sisäänkirjautumispalvelua tarjotaan myös esimerkiksi Silverlight- ja WP7-sovelluskehitysalustoille. C-Phone -sovellusta suunniteltaessa Windows Live -käyttäjähallinnan käyttö olikin luonteva valinta, koska sama päätös oli tehty jo aiemmin itse emosovelluksen kohdalla. Tällöin C-Phone -palvelua voi käyttää jo olemassa olevilla C-Care -tunnuksilla.

5 SOVELLUKSEN TOTEUTUS

Tämä osio on tarkoitettu pääasiassa ohjelmoijille. Sen vuoksi tämän otsikon alaotsikoissa oletan myös, että ohjelmointiin liittyvä perustermistö, kuten vaikka termit konstruktori ja metodi ovat tuttuja ja niiden käyttötarkoitus on myös sisäistetty. Koodi, johon tekstissä on viitattu, löytyy dokumentin perästä liitteistä.

5.1 MVVM-arkkitehtuurimallin toteutus

MVVM-arkkitehtuurin toteutuksessa näyttömallin luontiin on keskitettävä eniten huomiota ja se on toteutusta ohjaava pääelementti sisältäen tiedon ja näytön käsittelyn. Tiedon kulkeminen oikein näytön ja näyttömallin sekä näyttömallin ja mallin välillä ovat ratkaisussa oleellisessa asemassa, joskin myös näytöt ja mallit tulee toteuttaa huolellisesti.

Käytännössä näytön päivittäminen näyttömallista tapahtuu System.ComponentModel -nimiavaruuden ja INotifyProperty -luokkakirjastosta periytyvän päivittäjäluokan avulla, joka sijoitettiin pohjaluokaksi näyttömallipohjalle. Kaikki näyttömallit periytyvät tästä näyttömallipohjasta ja tällä pyritään vähentämään saman koodin toistuvaa

uudelleen kirjoittamista sekä näin myös nopeuttamaan sovelluskehitysprosessia. Näin myös mallit pystyvät periytymään suoraan tästä pohjaluokasta, jolloin myös ne voivat päivittää näyttöä sisältämättä kuitenkaan kaikkea näyttömallipohjan toiminnallisuutta, kuten esimerkiksi tietokantakontekstia. Liitteen kuvan 1 kuvateksti selventää vielä tämän Notifieriksi nimetyn apuluokan toimintaa ja itse kuva sisältää luokan C# ohjelmointikielellä toteutetun koodin.

Liitteen kuvassa 1 on kuvattu apuluokka, jonka avulla näyttömalleissa ja joissain tapauksissa myös malleissa muuttunut tieto päivitetään näytölle. Päivitys hoidetaan kutsumalla NotifyPropertyChanged -metodia, johon syötetään parametrina päivitettävän ominaisuuden nimi, joka on sidottu samalla nimellä näytön DataContextiin ja Binding -arvona näytölle. Liitteen kuva 2 havainnollistaa vielä muun muassa NotifyPropertyChanged metodin kutsumisen.

Paitsi ominaisuuksia kuten esimerkiksi merkkijonoja, päivämääriä tai lukuarvoja, niin näyttömallin kautta päivitetään näytölle myös komentoja samalla tavalla. Tämä toteutus on kuvattu liitteen kuvassa 2, jossa on komennon siirtämiseen näyttömallista näytölle käyttäjän käytettäväksi toteutettu koodipätkä. Tässä koodissa System.Windows.Input -nimiavaruuden sisältämä ICommand -luokkakirjaston tyyppinen AddCostCommand -ominaisuus päivitetään muuttuessaan näytölle liitteen kuvassa 1 kuvatun NotifyPropertyChanged -metodin parametrina. Tämä muutos tapahtuu luokan konstruktorimetodissa, jossa ominaisuuteen sijoitetaan AddNewCost() -metodi suoritettavaksi.

Edellä kuvattu näyttömalli tulee olla sidottuna XAML-näytön DataContext -ominaisuuteen oikealla tavalla, jolloin luokan ulkopuolelta kutsuttavaksi asetettua AddCostCommand -ominaisuutta voidaan kutsua XAML:sta suoraan käyttämällä DataContext -ominaisuutta hakevaa Binding -komentoa, kuten liitteen kuvassa 3 on tehty. Tässä kohtaa on huomattava, että ominaisuuksien nimeämisten tulee olla täsmälleen yhteneviä ulkoasultaan. Myös isot jo pienet kirjaimet tulee olla samanlaisia. Muuten datasidonta näytön ja näyttömallin välillä ei toimi. Liitteen kuvassa 3 HyperlinkButton -kontrollin Command -komentoon, joka toteutuu painettaessa hiiren vasenta näppäintä, on sidottu DataContext -ominaisuuteen sijoitetun näyttömallin sisältämä AddCostCommand -metodi.

Toteutus voi sisältää vielä esimerkiksi mallin käytön ja näytön kahden suuntaisen tiedon päivittämisen. Mallin käyttö toteutetaan sitomalla malli-luokka näyttömallin ominaisuudeksi, josta se päivitetään näytölle DataContext -ominaisuudeksi, jolloin tämän DataContextin alle sidotut kontrollit voivat kutsua malli-luokan julkisia olioita. Mikäli näytön Binding -arvo on asetettu kaksisuuntaiseksi, niin näytöltä tehtävät muutokset päivittyvät suoraan näyttömallin tai mallin sisältämään olioon. Oletuksena Binding -arvo kuuntelee vain DataContextista päin tulevia muutoksia yksisuuntaisesti.

5.2 Konversioapuluokat

MVVM-arkkitehtuuri toteutukseen voi myös toteuttaa olioiden tietotyyppiä muuntavia konversioapuluokkia, joiden avulla datan käsittely voidaan suorittaa vähemmällä ohjelmoinnilla toistettavalla koodilla. Konvertterien avulla voidaan esimerkiksi käsitellä Boolean -totuusarvoa näyttömallissa ja siirtää se sellaisenaan konvertterin kautta näytölle ilman ylimääräisiä koodirivejä näyttömallin puolella. Tällä tavalla koodista saadaan helpommin luettavaa, toistettavaa ja siten jatkokehitys onnistuu kivuttomammin. Konvertterin käyttö on esitelty tarkemmin liitteen kuvissa 4, 5 ja 6.

5.3 Windows Azure

Kertauksena tiivistelmä luvusta 4.2: Koska C-Phone -sovellus käyttää SQL Azure -tietokantaa eikä tietokantaan voida luoda suoraa yhteyttä, on luotava Windows Azure -palvelu. Näin WP7-sovellus ottaa WCF-palvelun kautta yhteyden tietokantaan. Tässä projektissa perustettiin tällainen palvelu ja ladattiin se jo olemassa olleelle Windows Azure -käyttäjätilille, jonka kautta myös sovelluksen käyttämä SQL Azure -tietokanta on tallennettu palvelimelle.

Käytännön toteutuksessa ennen kuin voidaan perustaa pilvipalveluna toimiva instanssi, tarvitaan pilvipalveluprojekti. Tämän projektin kautta sovelluskehittäjä hallitsee instanssia. Käytin tämän Windows Azure -projektin perustamiseen Microsoft Visual Studio ohjelmaa, johon on asennettuna Windows Azure Tools -lisäosa. Tämä yh-

distelmä tarjoaa valmiin projektipohjan Windows Azure -alustalta ajettaville sovelluksille. Projektipohja löytyy Visual Studiosta uutta projektia luodessa valitun koodikielen (C#, Visual Basic) alta Cloud -valikosta. Tarvitset myös tunnukset Windows Azure -portaaliin (http://manage.windowsazure.com) julkaistaksesi projektin. (Betts 2010, 121; Chappell 2010, 17-18.)

5.3.1 Projektin perustaminen Visual Studio 2010 apuna käyttäen

Luodessasi projektia Visual Studio generoi automaattisesti projektipohjan, joka sisältää ServiceConfiguration.csfg ja ServiceDefinition.csdef -tiedostot sekä Roles nimisen kansion, johon luodaan linkit projektiin lisättäviin työskentelijä- ja/tai web-rooleihin. Pystyt muokkaamaan ServiceConfiguration.csfg ja ServiceDefinition.csdef -tiedostoja joko suoraan muokkaamalla niiden sisältämää XML-sisältöä. Tätä kautta pystyt asettamaan muun muassa liitäntätunnisteet tallennuspalveluihin ja tuotannossa ajettavien rooli-instanssien määrän. Käytännössä mitä enemmän instansseja valitset, sitä nopeammin sovellus toimii, mutta instanssien käyttö on myös maksullinen ominaisuus. (Betts 2010, 122.)

Luotuasi projektin Visual Studioon voit lisätä ratkaisuun (*solution*) web- ja työskentelijärooleja, joihin voit luoda logiikkaa tai voit luoda lisäksi esimerkiksi Microsoft Silverlight 5 -projektin ratkaisuun ja linkittää sen web-rooliin. Työskentelijäroolissa voit esimerkiksi luoda yhteyksiä Windows Azure -tallennuspalveluihin. Mikäli sovellusta tarvitsee testata kehitysvaiheessa paikallisesti simuloidussa pilvipalveluympäristössä, tarvitset myös Windows Azure Development Fabric -ohjelmakomponentin. C-Phone -tapauksessa loin pelkästään web-roolin, joka sisältää WCF-palvelun. Tämä sen vuoksi, että toistaiseksi Windows Phone 7.1 -projektia ei voi linkittää suoraan web-rooliin, kuten Silverlight -projektin kanssa toimitaan. Näin saavutetaan kuitenkin kaksisuuntainen CRUD-yhteys (Create, Read, Update, Delete eli tiedon luonti, luku, muokkaus ja poisto toimenpiteet) tietokantaan. (Chappell 2010, 17-18.)

5.3.2 Instanssin perustaminen ja hallinta

Kun sovellus on valmis julkaistavaksi, sen siirtäminen Windows Azure -pilvipalvelujärjestelmään tapahtuu kahdessa vaiheessa. Ensimmäiseksi sovellus ladataan verkkopalvelun ohjausalueelle (*staging area*), josta sen voi siirtää tuotantoon. Ohjausalue eroaa tuotantopuolesta siten, että ohjausalueella ollessaan sovellus on saatavilla muodossa <GUID>.cloudapp.net. <GUID> on automaattisesti generoitu 32 merkkiä pitkä lukusarja, joka muodostuu kirjaimista ja numeroista. Tämä lukusarja vaihtuu aina instanssia päivitettäessä, joka tekee sovelluksen tehokkaan asiakaskäytön käytännössä mahdottomaksi. Tuotantoon siirretyn sovelluksen URL-osoitteen alkuosan voi taas määritellä itse ja se on pysyvä päivityksistä riippumatta. Näin sovellus on helpommin löydettävissä Internetistä. (Chappell 2010, 17-18.)

5.4 Versionhallinta

Sovelluskehityksessä versionhallinta on aina hyvä pitää mielessä eikä vähiten sovelluksen kehitystyötä silmällä pitäen. Tässä projektissa luonteva versionhallintaohjelmistoratkaisu oli Microsoftin Team Foundation Server (lyhyesti TFS), joka on suunniteltu käytettäväksi yhdessä Visual Studion kanssa ja se toimii yhdessä myös Expression Blend ohjelmaan ladattavana lisäosana. Noilla yhdistelmillä sovelluskehittäjä pystyy varaamaan tiedoston käyttöönsä tekemänsä muokkauksen ajaksi ja kun muokkaus on valmis ja testattu. Hän pystyy siirtämään muokatun tiedoston suoraan TFS:n käyttämälle palvelimelle, jolloin se on muiden kehittäjien edelleen käytettävissä sekä myös päällekkäiset, samanaikaiset että eri kehittäjien tekemät muutokset saadaan yhdistettyä Team Foundation Serverin mukana tulevalla Merge Tools apuohjelmaa käyttämällä. (Järvinen 2012, 30-32.)

Team Foundation Serverin muita hyödyllisiä ominaisuuksia on myös koodikirjastojen varmuuskopiointi palvelimelle, jolloin sovelluksen kehitystyötä voidaan jatkaa laiterikoista huolimatta. Muista kuin suoranaisesti versionhallintaan liittyvistä lisätoiminnoista mainittakoon TFS-tuki työkorttien lisäämiselle, jota tässäkin projektissa käytettiin. Työkortti (*work item*) on sähköinen lomake esimerkiksi vaatimusmäärittelyjen,

ohjelmavirheiden (*bug*) tai tehtävien (*task*) tekemiseen. Työkortille voi asettaa suorittajan ja vaiheen, esimerkiksi aloitettu tai valmis. Sovelluskehitystiimin kannalta on kätevää saada työkortit ja niihin liittyvä dokumentaatio käsille versiohallinnasta samaa ohjelmaa käyttäen, jota käytetään myös itse kehitystyöhön eli Visual Studiosta. (Järvinen 2012, 30-32.)

6 C-PHONE

6.1 Sovelluksen testaus

WP7-sovellus voidaan myös asentaa WP7-älypuhelimeen käyttämällä Windows Phone SDK -asennuspaketin RTM-version mukana tulevaa Windows Phone Application Deployment -työkalua. Tällöin tarvitaan Silverlight -projektin automaattisesti luoma .XAP-tiedosto, joka asennetaan Windows Phone -sovelluskehittäjälle rekisteröidylle päätelaitteelle. Näin pystyt testaamaan sovellusta samanlaisena, kuin se tulee toimimaan Windows Phone App Marketplacesta ladattuna. (WindowsPhoneGeek 2011.)

Tällä tavalla yritys voi myös käyttää sovelluksen kehitysversiota esitellessään sovellusta asiakkaalle. Lisäksi Windows Phone Application Deployment ohjelman kautta voidaan ajaa .XAP-tiedostoa WP7-emulaattorin kautta, jolloin sovelluksen demo onnistuu myös ilman WP7-päätelaitetta. (WindowsPhoneGeek 2011.)

6.2 Suunnittelun lähtökohta ja tavoite

C-Phone perustuu C-Care -tuoteideaan ja prototyyppiversiossa vielä etenkin sen tehtävänhallinta- ja tuntisyöttönäkymiin. Täten käyttöliittymää suunniteltaessa oli aluksi otettava nämä ominaisuudet tarkempaan tarkkailuun, jotta käyttötapaukset saatiin havainnollistettua. Käyttötapauksista ensisijaisen tärkeäksi valittiin tuntien mahdollisimman sujuva ja nopea kirjaus. Käyn kuvakaappausten avulla nämä C-Care -toiminnot tarkemmin läpi liitteen kuvissa 7 ja 8.

Liitteen kuvassa 7 käyttäjän on tarkoitus pystyä luomaan tai muokkaamaan tuotekohtaisia töitä, joiden perustietoihin kuuluvat muun muassa otsikko, tilaaja, työnumero, työtyyppi, työn vastaanottaja ja työn valmiusaste. Lisäksi työlle voi lisätä tehtäviä, joille kirjataan esimerkiksi tunteja tai hallitaan tarkastuslistoja viitetietona. Itse tehtävän perustietoihin kuuluvat otsikko, kuvaus, tehty toimenpide, tehtävän vastaanottopäivämäärä, kuin myös suunnitellut aloitus- ja valmistumispäivät, työmääräarvio tunteina, tehtävän suorittaja, työtyyppi, palvelutaso, elinkaari, turvallisuusohje, virheluokka, näkyvyys aikajanalla sekä tehtävän valmiusaste (valmis tai keskeneräinen).

Liitteen kuvan 8 tuntisyöttönäkymässä valitaan tuntien suorittaja reunuksen vasemmasta yläkulmasta olevasta ComboBox -kontrollista. Sen jälkeen selataan kalenterikontrollista haluttuun päivämäärään, johon syötetään tyypitettyjä tunti- tai kilometrikirjauksia, jotka viedään myöhemmin laskutukseen ja merkataan hyväksytyiksi. Reunuksen oikeassa laidassa olevalta listalta voi seurata henkilön suorittamia työtunteja valitulle tehtävälle.

6.3 Toteutuksen esittely ja sovelluksen käyttö

C-Phone -sovelluksen navigointi perustuu liitteen kuvassa 9 esiteltyyn pivot-näyttöön, jossa valitaan kohde, jonka tietoja hallitaan. Tämä tapahtuu valitsemalla ensin yritys, jonka alta valitaan tuoterakenne. Kolmannella pivot -sivulla valitaan valitusta tuoterakenteesta kohde, jonka tietoja kuten tuntisyöttöjä muokataan.

Kuten liitteen kuvassa 9 on kohdassa 1 valittu listalta yritys ”My Home”, jonka alta löytyvistä tuoterakenteista on kohdassa 2 valittu ”My Car” -tuoterakenne. Kohdassa 3 kolmannelta välilehdeltä on valittuna ”Engine” kohde, jonka alta löytyy 5 lapsiobjektia ja niiden alla olevista napeista voi siirtyä esimerkiksi kohteen töiden tai ominaisuuksien hallintaan.

Navigoitaessa esimerkiksi ominaisuuksien hallintaan avautuu uusi pivot-kontrolli, jonka ensimmäisellä sivulla näkyy lista valitun kohteen ominaisuuksista. Valitsemalla

listalta ominaisuuden, sen tietoja voidaan muokata toisella sivulla. Kolmannella sivulla voidaan lisätä kohteelle uusi ominaisuus.

Toisen ja kolmannen sivun ”Save”-nappia painettaessa päivitetyt tiedot päivittyvät tietokantaan ja listalle. Sivun yläreunassa käyttäjä näkee kokoajan valitun kohteen. Liitteen kuva 10 havainnollistaa tapahtumien kulun. Navigointi takaisin aloitusnäytölle tapahtuu WP7-standardin mukaan, jokaisesta WP7-puhelimesta löytyvällä ”Back”-näppäimellä.

Lisäksi C-Phone -sovellukseen kuuluu tuotteen identifiointi kuvalla, jonka voi napata suoraan WP7-älypuhelimen kameralla tai laitteelle tallennetuista kuvista. Sovellus käyttää samaa blob -tallennusarkistoa kuville kuin C-Care -emosovellus, joten älypuhelimessa pystyy näin käyttämään samoja tuoterakenteita kuin emosovelluksessa. (Liite: Kuva 11.)

C-Phone -sovelluksesta löytyy myös tehtävähallinta, johon navigoidaan myös tuotekohtaisesti aivan kuin ominaisuus- ja kuvahallintaan. Itse tehtävähallinnasta löytyy tuotteelle C-Care -emosovelluksessa perustetut projektit aikajärjestyksessä uusin ylimpänä ja kolmesivuisen pivot -näkymän seuraavalta sivulta löytyy projektille perustetut käynnissä olevat tehtävät C-Care -emosovelluksesta löytyvällä tehtävän elinkaaritiedon tunnusvärillä. Valitulta tehtävältä pystyy kuittaamaan sille luotuja tarkastuslistoja tehdyksi sekä seuraamaan ja luomaan uusia tuntikirjauksia. (Liite: Kuvat 12-14.)

Tarkemmin kuvattuna tuntisyöttö tapahtuu C-Phone -sovelluksen ”Hour-input”-näytöllä, jossa valitaan kirjattavan tunnin kirjaustyyppi, päivämäärä ja kirjattava tuntimäärä. Kun muutokset ovat tallennettu onnistuneesti, ne siirtyvät C-Phone -sovelluksessa ”Costs” -listalle ja hyväksytysti tallennetut työtunnit voidaan C-Care -emosovelluksessa siirtää edelleen laskutukseen. Kaikki tieto sovelluksessa on myös käyttäjätilikohtaista ja suojattu salasanalla. Nämä tilitiedot pitää saada yritykseltä, jotta sovelluksen käyttöönotto onnistuu. Tämä pätee sekä C-Care, että C-Phone -sovellukseen. (Liite: Kuvat 12, 13-15.)

7 PROJEKTIN TULOKSEN ARVIOINTI

Kun itse projektin kirjallinen tuotos lähenee nyt loppuaan, niin työn kirjoittajan roolissa huomaan, että paljon on tullut tehtyä. Toisaalta sovelluskehittäjänä näen myös, että paljon on vielä tekemättä, jotta sovellus olisi valmis ja käytettävä. Etenkin kun ottaa huomioon älypuhelinmaailman tarjoaman potentiaalin, niin jatkokehitystä tälle projektille kannattaa mielestäni ehdottomasti tehdä.

Itse projektia pidän myös onnistuneena ja uskon sen luovan hyvän pohjan jatkokehitykselle, joka olikin projektin alkuperäisessä tavoitteessa tärkeä kohta. Odotankin projektin eteenpäin vientiä myös innolla ja olen valmis kohtaamaan uusia haasteita tällä saralla. Vaikka mobiilikehitystä ei jatkettaisi Windows Phone -käyttöjärjestelmäspesifinä, niin projektissa tehtyä suunnittelutyötä voidaan jatkossa käyttää referenssinä myös muille mobiilialustoille toteutettaessa.

Monet kysymysmerkit älypuhelinsovellusten käyttö- ja kehitysmahdollisuuksista sekä yhteensopivuudesta jo aiemmin pystyssä olleen kehitysympäristön käyttämiin menetelmiin ovat ratkenneet. Esimerkkeinä mainittakoon vaikka SQL Azure -tietokannan käytettävyys kuten myös kirjautumispalvelun yhteensopivuus WP7-ohjelmointirajapintaan. Myös itse sovellukseen suunnitellut käyttötapaukset saatiin toteutettua. Markkinointimielessä sovelluksen toiminnan, vaikka se toistaiseksi vielä niukkaa onkin, nauhoittaminen videolle sekä emulaattorista että itse puhelimesta käsin tarjoaa yritykselle visuaalisesti vakuuttavan neuvotteluvaltin uusia mahdollisia asiakkaita haaliessa.

Olen tyytyväinen myös kirjalliseen tuotokseen ja toivon sen edelleen auttavan myös muita ihmisiä pääsemään sisään tässä projektissa käytettyihin sovelluskehitysmenetelmiin tai miksei myös tuotteen elinkaarihallintaan pyrkivään ajatteluun esimerkiksi. Tämä projekti onnistui hyvin herättämään mielenkiintoni alaa kohtaan ja yrityksen näkökulmasta prototyyppi- ja pilottihankkeena toteutus avasi suuntaa mobiilikehitysmaailmaan ja laiteriippumattoman palvelun kehitykseen.

LÄHTEET

Betts, D., Densmore, S., Dunn, R., Narumoto, M., Pace, E. & Woloski, M. 2010. Moving Applications to the Cloud on the Microsoft Azure Platform (Patterns & Practices). Microsoft Press

Cameron, R. 2011. Pro Windows Phone 7 Development. Springer Verlag

Chappell, D. 2010: Introducing Windows Azure. Viitattu 16.4.2015. Saatavissa: http://www.davidchappell.com/writing/white_papers/Introducing_Windows_Azure_v2.0-Chappell.pdf

Chuvyrov, E. & Lee, H. 2011. Beginning Windows Phone 7 Development 2 ed. Springer Verlag

Gren, M. 2008: HTML Password Box with Silverlight. Viitattu 16.4.2015. Saatavissa: http://www.codeproject.com/Articles/27408/HTML-Password-Box-with-Silverlight

Foley, M-J. 2010: Silverlight 4 to launch April 13. Viitattu 16.4.2015. Saatavissa: http://www.zdnet.com/blog/microsoft/silverlight-4-to-launch-april-13/5827

Heikniemi, J. 2009: Mitä uutta Silverlight 3:ssa?. Viitattu 4.5.2012. Saatavissa: http://www.itpro.fi/asiantuntijaryhmat/ohjelmistokehitys/Lists/Posts/Post.aspx?ID=199

Hyyppä, M. 2011: Satakuntalaisia pilvipalveluja. Viitattu 16.4.2015. Saatavissa: http://www.cieltum.com/upload/julkaisut/satakunnan-kauppakamari---cieltum.pdf

Järvinen, J. 2007: WPF/E on nyt nimeltään Silverlight. Viitattu 16.4.2015. Saatavissa: http://blogs.msdn.com/b/pasim/archive/2007/04/16/wpf-e-on-nyt-silverlight.aspx

Järvinen, J. 2012: Windows Phone sovelluskehitys, Docendo

Laitila, T. 2011: Microsoftin seuraava Silverlight-julkaisu jäämässä viimeiseksi? Viitattu 16.4.2015. Saatavissa: http://fin.afterdawn.com/uutiset/artikkeli.cfm/2011/11/09/microsoftin_seuraava_silverlight-julkaisu_jaamassa_viimeiseksi

Lehto, T. 2008: Microsoftin Silverlight 2 valmistui. Viitattu 8.4.2012. Saatavissa: http://www.tietokone.fi/uutiset/2008/microsoftin_silverlight_2_valmistui

Lehto, T. 2009: Silverlight 3 uudistaa web-sovelluksia ja videosivustoja. Viitattu 4.5.2011. Saatavissa: http://www.tietokone.fi/uutiset/2009/silverlight_3_uudistaa_web_sovelluksia_ja_videosivustoja

Masalin, T. 2010: Microsoftin Flash-haastaja uudistui. Viitattu 4.5.2012. Saatavissa http://www.tietokone.fi/softa/windows/microsoft_silverlight_4_0.

Microsoft A. Viitattu 16.4.2015. Saatavissa: http://www.microsoft.com/silverlight/what-is-silverlight/

Microsoft B. Microsoft Silverlight -tuen elinkaarikäytäntö. Viitattu 8.4.2012. Saatavissa: http://support.microsoft.com/gp/lifean45

Microsoft C. Windows Azure pricing overview. Viitattu 16.4.2015 Saatavissa: http://www.windowsazure.com/en-us/pricing/details/

Räihä, K. 2010: MIX10: Silverlight 4 RC nyt ladattavissa. Viitattu 16.4.2015. Saatavissa: http://fin.afterdawn.com/uutiset/artikkeli.cfm/2010/03/15/mix10_silverlight_4_rc_nyt_ladattavissa

Sæther, P. 2011: Using Model-View-Presenter (MVP) pattern in Windows Phone 7 projects. Viitattu 16.4.2015. Saatavissa: http://mobile.dzone.com/news/using-model-view-presenter-mvp

Sneath, T. 2007: Silverlight 1.1 is Now Silverlight 2.0. Viitattu 16.4.2015. Saatavissa: http://blogs.msdn.com/b/tims/archive/2007/11/29/silverlight-1-1-is-now-silverlight-2-0.aspx

Udell, C. 2007: Why Microsoft Silverlight Will Fail. Viitattu 16.4.2015. Saatavissa: http://visualrinse.com/2007/04/16/why-microsoft-silverlight-will-fail/

Wallaswaara, J. 2007: Silverlight 1.0 julkaistu ja Linux tuki tulossa. Viitattu 16.4.2015. Saatavissa: http://jukkawallasvaara.wordpress.com/2007/09/05/silverlight-1-0-julkaistu-ja-linux-tuki-tulossa/

WindowsPhoneGeek. 2001: Deploying WP7 XAP files to a device or emulator. Viitattu 16.4.2015. Saatavissa: http://www.windowsphonegeek.com/articles/Deploying-WP7-XAP-files-to-a-device-or-emulator

LIITTEET

```
using System.ComponentModel;

namespace PilotSilverlight.ViewModel
{
    public class Notifier : INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        protected void NotifyPropertyChanged(string name)
        {
            PropertyChangedEventHandler changedEventHandler = this.PropertyChanged;
            if (changedEventHandler != null)
            {
                changedEventHandler(this, new PropertyChangedEventArgs(name));
            }
        }
    }
}
```

Liite: Kuva 1. Notifier -apuluokka.

```
private ICommand _addCostCommand;
public ICommand AddCostCommand
{
    get { return _addCostCommand; }
    set
    {
        _addCostCommand = value;
        NotifyPropertyChanged("AddCostCommand");
    }
}

public TaskRowViewModel()
{
    AddCostCommand = new DelegateCommandExp(x => AddNewCost());
    SaveCommand = new DelegateCommandExp(x => SaveChanges());
```

Liite: Kuva 2. Tiedon päivittävä koodi näyttömallista näytölle.

```
<HyperlinkButton Style="{StaticResource navigationHyperlinkButton}" Command="{Binding AddCostCommand}">
```

Liite: Kuva 3. HyperlinkButton -kontrollin koodi, johon näyttömalli antaa kutsun.

```
public class TaskRowViewModel : ViewModelBase
{
    public bool RowExists
    {
        get { return _rowExists; }
        set
        {
            _rowExists = value;
            NotifyPropertyChanged("RowExists");
        }
```

Liite: Kuva 4. Kohdassa yksi TaskRowViewModel -näyttömalliin on sijoitettu totuusarvo tyyppinen RowExists –ominaisuus, joka muunnetaan näkyvyysarvoksi (*visibility*).

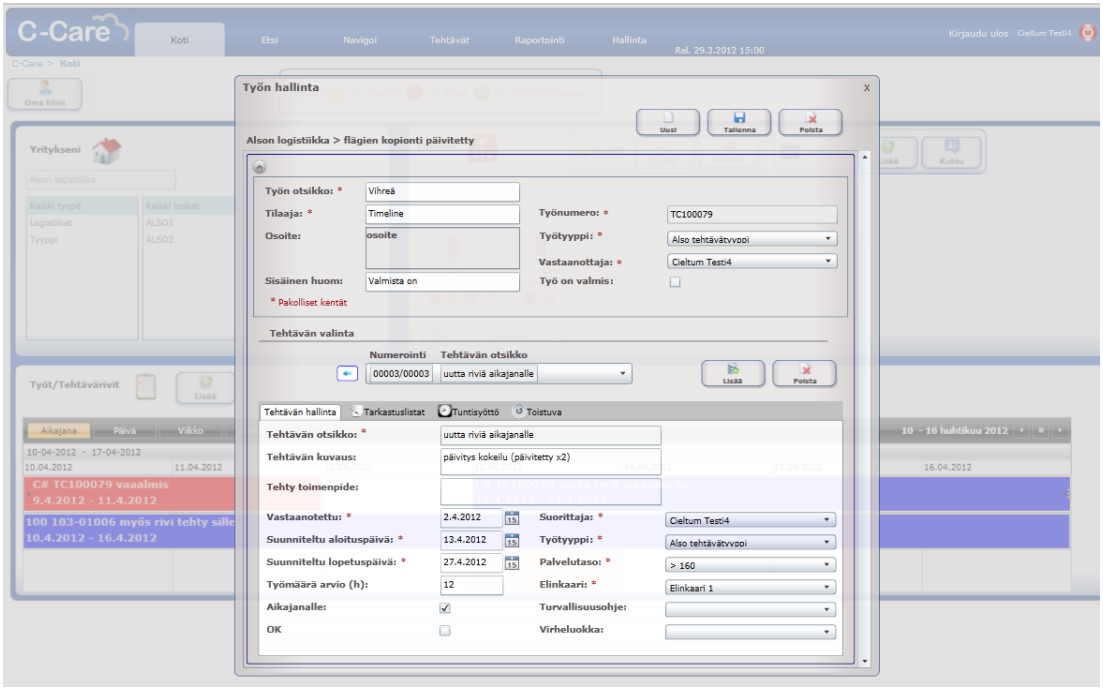
```
/// <summary>
/// Convert Controls Visibility property to bool and back
/// </summary>
public class BoolToVisibilityConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        var visibility = (bool)value;
        return visibility ? Visibility.Visible : Visibility.Collapsed;
    }

    public object ConvertBack(
        object value, Type targetType, object parameter, CultureInfo culture)
    {
        var visibility = (Visibility)value;
        return (visibility == Visibility.Visible);
    }
}
```

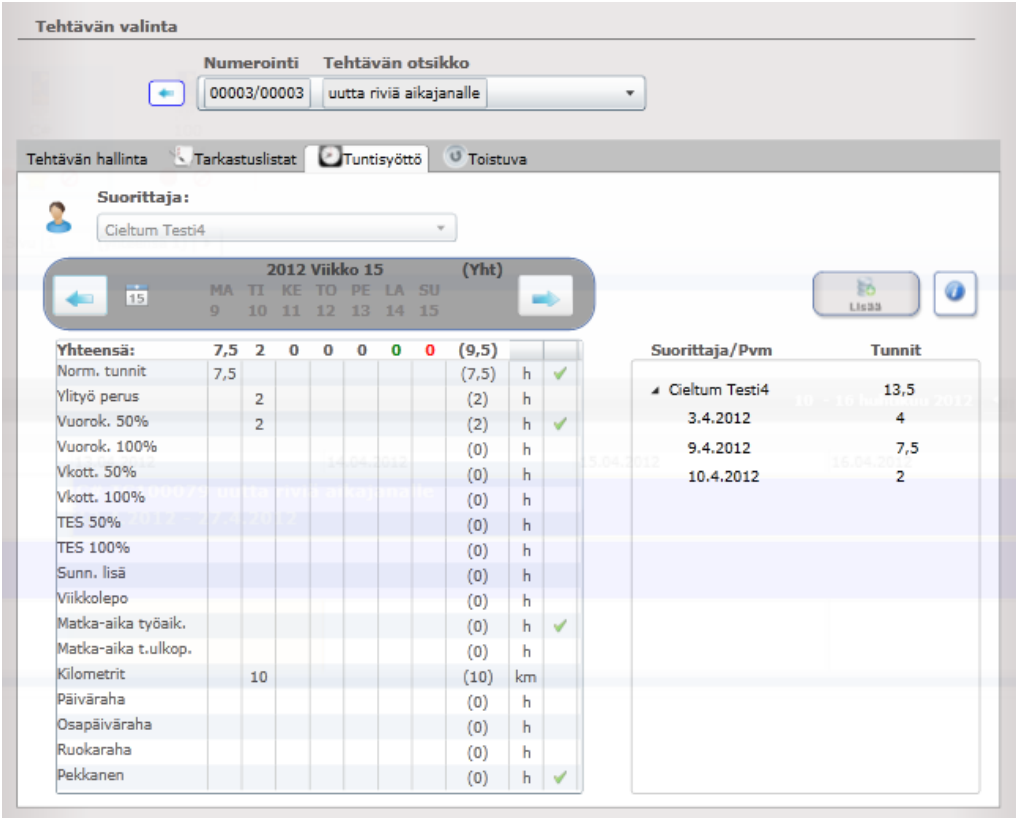
Liite: Kuva 5. Totuusarvon näkyvyys arvoksi ja toisin päin muokkaama apuluokka.

```
<phone:PhoneApplicationPage.Resources>
    <Converters:BoolToVisibilityConverter x:Key="bvConverter" />
</phone:PhoneApplicationPage.Resources>
<StackPanel Visibility="{Binding RowExists, Converter={StaticResource bvConverter}}"
```

Liite: Kuva 6. TaskRowViewModel -näyttömallia käyttävän näytön resursseihin on lisätty konvertteri ja sijoitettu näyttömallin sisältämä RowExists -ominaisuus konvertterin kautta käännettynä Binding -arvona näytölle.



Liite: Kuva 7. Kuvakaappaus C-Care -sovelluksen tehtävähallinta lapsi-ikkunasta 10.4.2012.



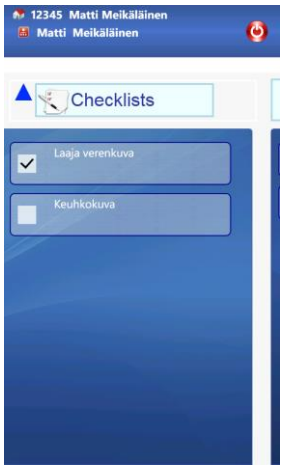
Liite: Kuva 8. Kuvakaappaus C-Care -sovelluksen tehtävähallinta lapsi-ikkunan tuntisyöttö välilehdestä 10.4.2012.

Liite: Kuva 9. C-Phone -sovelluksen aloitusnäkymä.

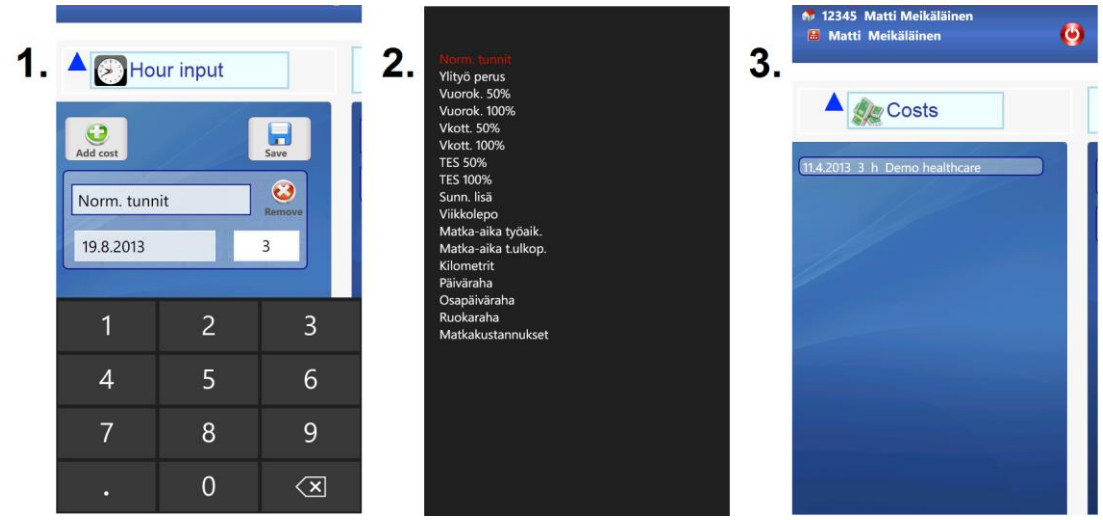
Liite: Kuva 10. C-Phone -sovelluksen tuotteen ominaisuuksienhallintanäkymä.

Liite: Kuva 11. C-Phone -kuvahallintanäkymä.

Liite: Kuva 12. C-Phone -tehtävienselausnäkymä.



Liite: Kuva 13. C-Phone -tarkastuslistojenkuittausnäkymä.



Liite: Kuva 14. C-Phone -tuntisyöttönäkymiä.

Liite: Kuva 15. C-Phone -sisäänkirjautumisnäkymä.